Fixing the 2006 Federal Voting Standards

Earl Barr, Matt Bishop, Mark Gondree

Introduction

In elections across the nation, electronic voting machines have failed to boot, tallied ten times as many votes as registered voters, and drawn criticism from academics, election officials, and concerned citizens alike. High-profile exploits, such as the Hursti attack [4] and the Princeton group's Diebold virus [2], have brought media attention to the fragility lurking below the surface of voting systems. In light of these problems, it is perhaps an understatement to say that election systems have flaws. Yet independent testing authorities certified these very systems as meeting explicit Federal standards for electronic voting machines. What went wrong?

To date, researchers have focused on two explanations: these systems were poorly designed and the certification process is flawed. Concerning design, researchers have shown, and experience has confirmed, that electronic voting machines do not meet reasonable expectations for correctness, availability, accessibility, and security. A large body of work proposes immediate, short-term "fixes," but, in each case, finds the only long-term remedy to be a complete system redesign. Concerning certification, researchers have pointed out that the practice of having the voting machine vendors pay the independent testing authorities raises questions about the impartiality and rigor of the certification process itself. They have also decried the fact that the certification process inhibits the incremental improvement of the system tested by focusing on whether the system passes, and not on what the vendor could do to improve the system.

We contend that the problem is still more fundamental: the standards on which vendors necessarily base their system designs and against which the testing authorities certify those systems are flawed. These standards, promulgated by the Federal Election Commission (FEC) and now the Election Assistance Commission (EAC), do not express a coherent set of requirements for electronic voting systems. They contain no system model or threat model. Lacking these guides, any standard will be a patchwork of ideas and requirements that fails to achieve its goals — if, indeed, those goals are even clear. Without clear requirements, no design can be sound nor can any system be meaningfully certified.

Neither the government nor electronic voting system vendors have adequately addressed these design and certification flaws, let alone advanced solutions as national standards, for the simple reason that their effort has been misdirected. They need our community's help and active engagement in the writing of clear, sound, and testable standards.

In this paper, we first investigate the lacunae of the current federal voting system standards, before making a number of recommendations for how to improve them. These recommendations do not call for mere additions to the standards, but highlight unanswered questions in applying system, threat, and process modeling to secure systems' standards, and research opportunities. Improving the standards is the first step toward improving the quality of voting systems. The goal is, of course, electronic voting machines that deliver on their promise and strengthen our democracy.

The Federal Voting Machine Standards

In October 2002, Congress passed the Help America Vote Act (HAVA). The act created the EAC, whose duties include the development of a set of voluntary guidelines that states can use to be sure their equipment meets the (rather ambiguous) HAVA requirements. In essence, Congress gave the EAC the duty of making these requirements formal and precise, so they are meaningful to vendors and state election officials. HAVA also established several important timelines, including the Jan 1, 2006 deadline by which states were to meet the aforementioned federally-mandated improvements to voting systems. As of August 2005, $2.5 billion had been dispersed to the states, territories, and the District of Columbia to these ends.

The guidelines promulgated by the EAC "provide a set of specifications and requirements against which voting systems can be tested to determine if they provide all the basic functionality, accessibility and security capabilities required to ensure the integrity of voting systems." Although these guidelines apply to voting systems in general, this paper considers them in the context of electronic voting systems such as DRE (Direct Recording Electronic) systems and optical scanning recording systems for paper ballots. However, much of our discussion generalizes readily to other types of voting systems as well.

The EAC commissioners were appointed in December 2003. The EAC received $1.2 million in funding for fiscal year 2004, and $14 million in fiscal year 2005. During this period, it released a draft of the "Voluntary Voting System Guidelines" and opened it to a 90-day public comment period. At the end of 2005, the EAC released its official version of these standards. Our criticisms here reference

the 2005 EAC standards, but should be recognizable to those familiar with the earlier standards and drafts; they all mostly overlap in content.

Problems with the Standards

In a broad sense, the inadequacy of the standards is not surprising. A basic impossibility result, Rice's theorem, states that no single check-list or algorithm can guarantee that an arbitrary computer, equivalent to a Turing machine, satisfies a nontrivial property such as correctness and security. In practice, check-lists can increase the likelihood that a system meets some nontrivial property only when the class of systems to which they apply consists of computers of similar architecture, configuration, and used with similar procedures in similar environments. When DREs are built on general-purpose computers and the environments in which they are used differ widely, as is currently the case, even the assurance a check-list can provide is undermined.

The way the standards fail *in practice*, however, is far more basic. The standards fail to make precise the usability requirements outlined by HAVA. They confuse requirements for accurate voting with requirements for simplifying system testing. They include seemingly arbitrary specifications; for example, the acceptable error rates (Vol. I, sec. 3.2.1) seem to have been chosen arbitrarily. They mandate impossible features; for example, they require that shared resources do not leak information (Vol. I, sec. 7.5.4), but there is no way to prevent this — shared resources can *always* be used as a covert channel, although it is possible to reduce this channel's effectiveness. In short, the standards seem vague, ad hoc, arbitrary, and sometimes unreasonable.

Making the System Publicly Available

Critics of current voting systems frequently point out that they are proprietary and therefore voters cannot inspect source code to ensure that the software works correctly. While true, this complaint actually reflects a larger issue — that the system design and implementation is known only to the manufacturer.

Consider the software of the electronic voting machine. Saltman [7] first cited hidden malicious code as a problem in electronic voting. Simons [8] describes locating such code as "akin to finding the proverbial needle in a haystack." Dill [1] goes further, claiming it is "much harder than finding a needle in a haystack." The security community has recognized this as such a clear problem that it has become the subject of exercise and recreation: several university computer science courses assign a "Hack-A-Vote" homework to graduate students [10, 6], and the "Obfuscated V" contest turns the same concept into a game [3]. The problem is well-known in other areas of computer security. For example, Thompson [9] described how to place a Trojan horse into a compiler in such a way as to provide illicit access to a system, yet be undetectable unless the executables produced by that compiler were analyzed. The moral is that one can never verify that a system has no flaws, not even if *all* source code is available.

This state of affairs reflects reality. We live with imperfect systems. Cars break down. Billing systems malfunction. Electoral processes are susceptible to problems: recall, for example, the electoral corruption of New York City during the era of Tammany Hall. Perfect voting systems do not exist. The goal is to build voting systems that are as good as possible.

To this end, making the source code available for public inspection enables voters who have the desire and expertise to analyze that code base for flaws. It is analogous to the ability to observe all aspects of a paper-based election, except, of course, an individual marking her votes, to preserve the secrecy of her ballot. In a paper election, an observer may observe the ballots being counted. But many electronic voting systems currently being used[1], record votes and ballot images as bits on flash cards and in memory. They tally these votes and report totals. An observer cannot look inside the memory to verify that the ballots are recorded correctly or the votes are tallied correctly because the bits are not visible. Analyzing the source code and system can increase confidence that the votes are recorded correctly.

Assume that the source code for the electronic voting systems, their operating systems, and all ancillary source code is available for public inspection. Presumably, experts will examine it, as researchers at the Johns Hopkins and Rice University did when source code purported to be that of Diebold became publicly available. While this would uncover many *software* flaws, source code analysis alone will not provide the level of assurance commensurate with the goals of electronic voting because those goals involve policies and procedures as well as software assurance.

Software, and computer systems, exist in an environment with policies and procedures. Unless these policies and procedures are taken into account, reviewing only the software may give a misleading idea of the security of a system. For example, using an SSL-like protocol to provide confidentiality of precinct vote data [5] as it is sent to the central counting system sounds like good security, but is irrelevant. Election officials announce precinct vote data when they announce final results of the election. The transmission of that data requires integrity and mutual authenticaion, not confidentiality[2]. So, even though the software might appear to a reviewer to provide the requisite security in transferring precinct vote data, it does not, since the confidentiality it provides is not a requirement.

Software security review is not enough. Even if the software is created using very high assurance techniques, we must still ask: What are we assuring that the software will do? Without meaningful standards, this question remains puzzling.

Threat and System Models

The most striking deficiency of the standard is the lack of an answer to the question "against what threats should the system be protected?" In many places, the standards presume an implicit system model (e.g., polling places and central tabulating facilities both exist, are physically separate, and data must be sent between them), without specifying any associated design requirements. Because it is implicit, the model is so unclear that we have no language with which to discuss the threats, and the various players — election officials, testers, and vendors — are left to their own devices. For example, the standards imply roles, such as installer and troubleshooter, for running the electronic voting machines at precincts, but do not articulate the details of the implied roles. So access control policies, which manage how election staff can interact with the system, are left entirely to the vendor's discretion. The pertinent requirement that the access control policy "provide effective voting system security" is, again, vague and untestable, and fails to

[1] Some jurisdictions use electronic voting systems as ballot marking devices. When used in this manner, the systems do not record votes electronically, and paper ballots are counted.

[2] When used properly, SSL allows the client to authenticate the server and send integrity-checked messages but it does not provide mutual authentication. For that, SSL requires a public key infrastructure or other key distribution mechanism.

identify the corresponding threats that it is meant address.

Only with the knowledge of a threat can the sufficiency of a countermeasure be ascertained. Only against a description of a system (a model) can our language about processes and procedures be meaningful. With neither, the standards communicate only an ad hoc list of requirements that are open to misinterpretation.

Lets return to the example of data integrity during transmission. The section *Telecommunications and Data Transmission* (Vol. I, sec. 7.5) addresses this: "[v]oting systems that use electrical or optical transmission of data shall ensure the receipt of valid vote records is verified at the receiving station." Verification at the receiving station is necessary, but clearly insufficient. A *man-in-the-middle attack* takes advantage of just this type of scenario, where verification is not performed by both parties. Consider this simple example: Bob expects a call from Alice on a public phone. Earlier, they arranged a secret codeword only they know. Bob, upon picking up the phone, will not speak until he hears this codeword. Alice, however, has no way to verify that Bob has picked up the phone instead of, say, Mallory. By diverting or intercepting Alice's call, Mallory can hear the secret and use it for later impersonation. Clearly, one-way authentication is insufficient to ensure data integrity.

That same section (Vol. I, sec. 7.5) also requires appropriate measures be taken to detect an "intrusive process" which may intercept the data. Does this mean a process running on the same computer (like a Trojan horse), or a process running on the network (like a port scanner), or a physical process that might intercept data (like a wiretap)? Previous versions of the standards alternate between each of these notions, but the current standard leaves "intrusive process" undefined. Furthermore, the implications of this requirement are unclear: if interception is to be detected, does that ban technologies where interception cannot be detected, such as a broadcast medium like Ethernet or a wireless medium?

As another example, consider the target error rate of *Accuracy Requirements* (Vol. I, sec. 4.1.1), which states that the voting system "shall achieve a target error rate of no more than one in 10,000,000 ballot positions." The reason for the number 10,000,000 is not given; as far as the reader can tell, the required rate could just as well be no more than one in 500,000 ballot positions, or no more than one in 100,000,000 ballot positions. In fact, these rates are all considerably lower than the error rates of paper ballots. The effort to formulate a system model would centralize and bring the question of acceptable error rates to the fore, and thereby help determine which of these numbers provides the desired level of security and accuracy.

The requirements in *Protecting Transmitted Data* (Vol. I, sec. 7.7.3) pose both rationale and interpretive problems. The standards require that "transmitted data ... needs to be protected to ensure confidentiality and integrity" and cites ballot definitions, precinct counts, and the opening and closing of poll signals as examples of information that needs to be protected. As pointed out earlier, though, the first is displayed on the voting machines themselves, the second is published as part of the reporting of election results, and the last two are observable actions. So the reason they must be confidential during transmission is unclear. The interpretation of part b, which says the "capability to transmit non-encrypted and non-authenticated information via wireless communication shall not exist" can be read as allowing the system to have wireless hardware that is kept turned off, or as disallowing that hardware altogether. The interpretation depends on the threat model. If the model assumes that people with the ability to access and enable the wireless hardware will not do so while the system is in use, then the former interpretation is acceptable; but if insiders are considered threats (either because they are untrustworthy or because they may accidentally enable, or leave enabled, the wireless hardware), then the latter interpretation is necessary.

In short, incorporating system and threat models into the standards will make clear the purpose and reasons for certain aspects of the standards that are currently nebulous. As a final and archetypal example, consider that the standards currently insist a voting machine pass a "system test" before operation. This has led some vendors to write a simple program that prints "System Test Passed" on start up. No one knows what a "system test" actually is or what it is meant to defend against, but it is clear the machines will not be certified without this message.

Conclusion

The failure of the standards to include a threat model and a system model makes certification of voting systems haphazard at best. It increases the difficulty of developing, testing, and certifying systems.

Rewriting the standards to include a threat model would enable vendors, election officials, computer scientists, and in fact *all* citizens to see what threats the developers of the standards consider necessary to protect against. The testers would know what their tests needed to check, and additional parts of the standard could describe the limits of the testing.

The focus of work in electronic voting systems has been to detect and remediate problems in the existing systems. This work has been invaluable in gathering evidence of the sorry state of the art in this area, both in the systems and in the standards used to certify them. Now is the time to work on making these technologies, and the electoral process in which they are used, work at least as well as conventional, paper-oriented election process. In particular, open source election software is not enough! The transparency of ballot design, configuration for the systems (e.g., access controls), and procedures for running the election will determine whether observers can assess the fairness of this process, and the accuracy of the results. Further, both the standards and certification system must respond to feedback from election administrators and incorporate the results of relevant research (such as error rates and ballot design). Perhaps the model CERT/CC uses to disseminate information that affects the security of computer systems could be adapted here. Finally, we need to move beyond the "patch" approach to electronic voting systems, and instead use high assurance techniques to design and implement systems that provide the requisite guarantees.

As a community, we have much to contribute to the development of effective and meaningful standards for electronic voting machines. In concert with voting officials and the body politic, we can help secure these systems and, thereby, our rights as voters.

1 References

[1] D. Dill. Electronic voting: An overview of the problem. Talk for the Carter-Baker Commission on Federal Election Reform in Washington, D.C, Apr. 2005. http://www.verifiedvotingfoundation.org/article.php?id=5731.

[2] A. J. Feldman, J. A. Halderman, and E. W. Felten. Security analysis of the Diebold AccuVote-TS voting machine, Sept. 2006. http://itpolicy.princeton.edu/voting/.

[3] D. Horn. The Obfuscated V contest. http://graphics.stanford.edu/~danielrh/vote/vote.html, 2004.

[4] H. Hursti. Diebold TSx evaluation and security alert, May 2006. http://www.blackboxvoting.org/BBVtsxstudy.pdf.

[5] RABA Innovative Solution Cell. Trusted Agent Report: Diebold AccuVote-TS Voting System, Jan. 2004. http://www.raba.com/press/TA_Report_AccuVote.pdf.

[6] A. Rubin. CS600.443 project 2. http://www.cs.jhu.edu/~rubin/courses/sp04/proj2.txt, 2004.

[7] R. G. Saltman. Accuracy, integrity and security in computerized vote-tallying. *CACM*, 31(10):1184–1191, 1988.

[8] B. Simons. A response to: The league of women voters of the US questions and answers on direct recording electronic (DRE) voting systems, 2004. http://www.leagueissues.org/lwvqa.html.

[9] K. Thompson. Reflections on trusting trust. *CACM*, 27(8):761–763, 1984.

[10] D. Wallach. CS523 Hack-A-Vote project. http://www.cs.rice.edu/~dwallach/courses/comp527_f2003/voteproject.html, 2003.